Bigraphical Refinement*

Gian Perrone Søren Debois Thomas Hildebrandt

Programming, Logic and Semantics Group
IT University of Copenhagen
Copenhagen, Denmark
{gdpe,debois,hilde}@itu.dk

We propose a mechanism for the vertical refinement of bigraphical reactive systems, based upon a mechanism for limiting observations and utilising the underlying categorical structure of bigraphs. We present a motivating example to demonstrate that the proposed notion of refinement is sensible with respect to the theory of bigraphical reactive systems; and we propose a sufficient condition for guaranteeing the existence of a safety-preserving vertical refinement. We postulate the existence of a complimentary notion of horizontal refinement for bigraphical agents, and finally we discuss the connection of this work to the general refinement of Reeves and Streader.

1 Introduction

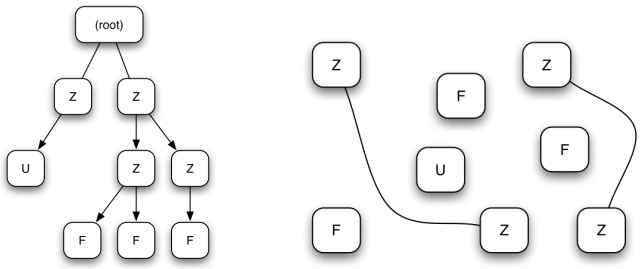
Refinement is the process of gradually developing a specification towards a suitable implementation, through a series of steps in which more concrete entities are shown to be as acceptable as the more abstract entities preceding it in the chain of refinement steps, based upon what may be observed of these entities. The utility of this method has been demonstrated through many years of application in academic and industrial settings. In this paper we attempt to bring these well-studied benefits to a new class of systems — namely, bigraphical reactive systems. We focus primarily on *vertical refinement* [3], where the aim is to relate models constructed with respect to different semantics.

A *bigraphical reactive system* [21, 19] (BRS) is a model construction paradigm proposed by Milner and colleagues that aims to enable modelling of interactive systems within a cohesive theoretical framework. While the primary long-term focus of bigraphs is on models of ubiquitous and context-aware systems [1], they have demonstrated value in other areas such as biological applications [15, 5, 6] and business processes [12, 25]. Bigraphical reactive systems also capture the syntactic and semantic structure of many formalisms associated with process modelling, providing a unifying meta-calculus within which to relate many of these well-developed theories. Already encodings into various bigraphical reactive systems have been demonstrated for amongst others the λ-calculus [20], CCS [19], the Mobile Ambients calculus [14], several variants of the π-calculus [14, 4, 8], Fusion Calculus [10] and Petri Nets [16].

Bigraphical reactive systems consist of two graphs (hence the name *bi*graph) modelling the orthogonal notions of *locality* and *connectivity* which together capture the static structure of a system, and a set of *reaction rules* that may selectively rewrite portions of the bigraph in order to capture the dynamic behaviour of that system. We will introduce bigraphs and bigraphical reactive systems (assuming no prior knowledge) in Section 2.

*This work funded in part by the Danish Research Agency (grant no.: 2106-080046) and the IT University of Copenhagen (the Jingling Genies project). The first author would like to thank Prof. Steve Reeves and Dr. David Streader for hosting him as a visiting researcher at the University of Waikato during the early stages of this work, and for helpful discussions during this time.

J. Derrick , E.A. Boiten, S. Reeves (Eds.):
Refinement Workshop 2011.
EPTCS 55, 2011, pp. 20–36, doi:10.4204/EPTCS.55.2



(a) Place Graph (b) Link Graph

Figure 1: The constituent place (1a) and link (1b) graphs that form a particular bigraph.

The usual notion of "observation" in a BRS is derived from the above notion of dynamic behaviour: a BRS gives rise to an LTS, the labels of which are simply the least context enabling reaction. The present effort towards refinement takes this connection between static structure and dynamic behaviour to heart, and attempts to short-circuit the LTS in favour of a more directly structural mechanism of refinement. This makes sense uniquely for bigraphs exactly because of the close correspondence between structure and dynamics. The primary contribution of this paper is to introduce such a mechanism as a small step towards bringing the well-established benefits of refinement to models constructed within the bigraph formalism. Additionally, we give a sufficient condition for an abstraction functor (Section 4) to give rise to a safe refinement, and show that this notion of refinement corresponds with (and indeed, in part is an instance of) the general refinement of Reeves and Streader [23, 24].

1.1 Structure of the paper

The remainder of this paper is structured as follows: We review bigraphs (assuming no prior knowledge) in Section 2. In Section 3 we introduce a running example that will be used to illustrate all of the concepts presented. In Section 4 we present our definition of vertical refinement for bigraphical reactive systems and show that the proposed refinement preserves safety properties with respect to the abstraction functor upon which it is parametrised. Additionally, we present a sufficient condition for an abstraction functor to give rise to a safe refinement. Finally, in Section 5 we discuss a candidate horizontal refinement mechanism for bigraphical agents, derived from the general refinement of Reeves and Streader [23, 24], and discuss the connection of this work to general refinement.

2 Bigraphical Reactive Systems

Bigraphical reactive systems is a graphical formalism emphasising the orthogonal notions of *locality* and *connectivity*. A BRS is a category of bigraphs and a set of reaction rules that may be applied to rewrite these bigraphs. We provide here a short, informal introduction to the anatomy of a BRS without assuming any prior knowledge. For a complete treatment of bigraphs and BRSs, readers are referred to [21, 19].

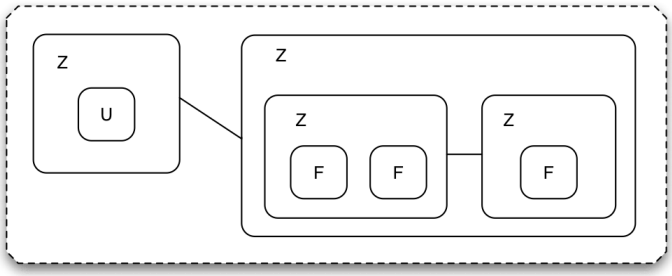


Figure 2: The bigraph resulting from the combination of the place and link graphs in Fig. 1a and Fig. 1b. This bigraph is an agent of the BRS_{notify} example BRS with signature $\Sigma = \{\mathsf{Z}, \mathsf{U}, \mathsf{F}, \mathsf{N}\}$ that we will introduce in Section 3.

2.1 Static Structure

The most basic construction within the static fragment of bigraphical reactive systems is the *node*. This follows from normal definition of a node within graph theory. To nodes we assign *controls*, which are drawn from a *signature* Σ, the set of controls. We sometimes use a convenient shorthand such that we may refer to a node as being an "X node", by which we really mean a node that has been assigned the control X. Nodes may be nested to arbitrary depth to form a tree that is known as the *place graph* (Fig. 1a). We represent this nesting by containment, as shown in Fig. 2. We distinguish between controls of two kinds: *active* and *passive* ones; we shall see later how active controls admit dynamic behaviour beneath them whereas passive controls do not. Every tree of nodes is contained by a *region* (the dotted border in Fig. 2). Bigraphs permit multiple regions (a place forest).

To controls (and therefore nodes) we assign a fixed *arity*, which defines the number of *ports* that a given node possesses. A port is a connection point on a node; it must always be connected to other such connection points by the *link graph*. The link graph (Fig. 1b) is an undirected hypergraph over the ports of the nodes of the place graph. A single (hyper) edge may connect arbitrarily many ports on different nodes.

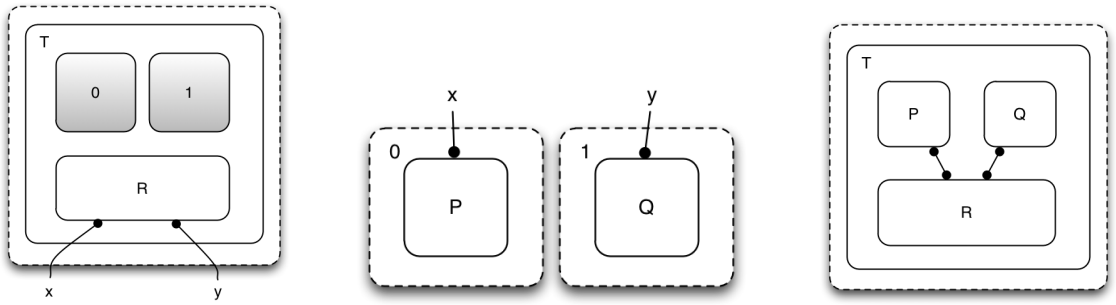
Within the place graph, in addition to regions and nodes, there may also exist *holes* (known as *sites* in some bigraphs literature), which are expressed visually as shaded grey nodes (as in Fig. 3a). A hole is a location into which a region of another bigraph may be inserted by composition. It may be helpful to think of bigraphs with holes as "contexts" and those without as "processes" or "terms".

Present also within Fig. 3 are *names* that represent (named) points at which edges of the link graph may be fused to form a single (hyper) edge. In the intuition of contexts and terms, names of bigraphs roughly correspond to unstructured names, as in the π-calculus. By convention, *outer names* are drawn upwards, and *inner names* are drawn downwards. Outer names are analogous in the link graph to regions in the place graph, while inner names are analogous to holes. Through composition of link graphs, sets of inner and outer names that agree are matched and joined.

Definition 1 (Interface)**.** *An interface is a pair $\langle j, X\rangle$ where $0 \leq j$, indicating the number of holes or regions, and X is a set of (inner or outer) names.*

Definition 2 (Bigraph)**.** *A bigraph is a 5-tuple:*

$$(V, E, ctrl, prnt, link) : \langle k, X\rangle \to \langle m, Y\rangle$$



(a) $A : \langle 2, \{x,y\}\rangle \to \langle 1, \emptyset\rangle$ (b) $B : \langle 0, \emptyset\rangle \to \langle 2, \{x,y\}\rangle$ (c) $A \circ B : \langle 0, \emptyset\rangle \to \langle 1, \emptyset\rangle$

Figure 3: The composition of two bigraphs A and B with their respective interfaces

Here V is the set of nodes, E is the set of hyperedges, $ctrl$ is the control map *that assigns controls (and therefore arities) to nodes, $prnt$ is the* parent map *that defines the tree structure in the place graph and $link$ is a link map that defines the link structure. The inner interface $\langle k, X\rangle$ indicates that the bigraph has k holes, and a set of inner names X. The outer interface $\langle m, Y\rangle$ indicates that the bigraph has m regions and a set of outer names Y.*

Definition 3 (Composition)**.** *Bigraphs are composed separately in the place and the link graphs. The interfaces of the bigraphs must be compatible in order for composition to be defined, i.e., the sets of names and the number of regions/holes must be the same. Fig. 3 illustrates the composition $A \circ B$ of bigraphs A and B. In the place graph, we insert contents of the left-most region of B into hole 0 of A, and the contents of the right-most region of B into hole 1 of A. Regions are numbered left-to-right: we insert the contents of region 0 into hole 0 etc. In the link graph, links are spliced together where there is name agreement between the inner and outer names of the bigraphs being composed. We may refer to A in this case as being a* context *into which B is inserted.*

Definition 4 (Tensor Product)**.** *There exists an additional way in which to combine bigraphs, namely the* tensor product *$A \otimes B$, where A and B are bigraphs. Where A and B do not share any inner or outer names, this just involves juxtaposing their place graphs, taking the union of their names, and increasing the indices of holes in B to make them unique with respect to A. This definition obscures some technical details. It is recommended that readers interested in following the proofs in Section 4.1 refer to [21] for a precise definition.*

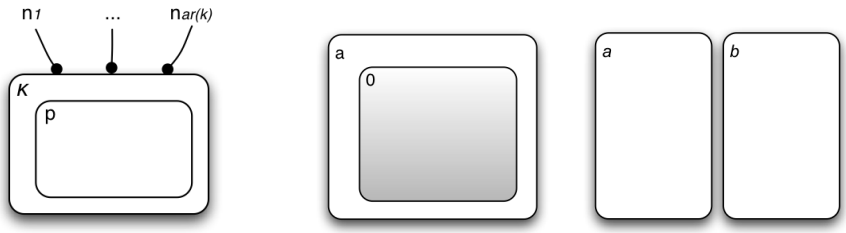
2.2 Notation

We introduce a rudimentary term language for representing bigraphs that should be familiar to most readers accustomed to the notation for process algebras. The present language is not complete, i.e., it cannot express every bigraph, but it can express the ones we will use in examples. It is a subset of a complete such language [18]. We will use this term language in conjunction with the graphical representation used in Fig. 2.

Definition 5 (Bigraph Term Language)**.**

$$p ::= \kappa(n_1, \ldots, n_{ar(\kappa)}).p \mid p \mid p \mid -_i \mid \mathsf{nil}$$

Where $\kappa \in \Sigma$.



(a) $\kappa(n_1,\ldots,n_{ar(\kappa)}).p$ (b) $a.-_0$ (c) $a.\mathsf{nil} \mid b.\mathsf{nil}$

Figure 4: Example bigraph terms with their associated graphical representation

The term language requires some explanation — $\kappa(n1,\ldots,n_{ar(\kappa)}).p$ is *prefixing* (Fig. 4a), indicating a node assigned the control κ. The arity of κ is given by $ar(\kappa)$. The sequence $n_1,\ldots,n_{ar(\kappa)}$ are the ports of the node. Finally, the suffix p is the term that is nested inside this node. $p \mid p$ is *juxtaposition* of terms (Fig. 4c), placing them as siblings within the place graph. $-_i$ is a hole (Fig. 4b), indexed by some integer $0 \leq i$. Finally, nil is the nil terminator which is simply the empty graph in the graph representation.

2.3 Dynamics

Having introduced the basic structure of *bigraphs*, the static portion of a BRS, we now introduce the *reactive* portion of a BRS that imbues a system with dynamic behaviour. This relies on *reaction rules* that define rewriting that may be applied to a bigraph. A reaction rule (R,R',η) consists of a *redex* R, a *reactum* R' and an *instantiation map* η, where the redex is a bigraph to be matched and the reactum is the bigraph with which the matched portion of the bigraph should be replaced. The instantiation map indicates how parameters matched by holes in the redex should manifest in the reactum after matching. Where the instantiation map is unambiguous (e.g., it is the identity map), we may just write $R \to R'$.

Definition 6 (Reaction)**.** *Matching of a particular reaction rule (R,R',η) against a particular bigraph G and rewriting it into some other bigraph G' proceeds by decomposition of the bigraph into a* context *C, a* match *R (the redex), and a set of* parameters *d (for portions of the bigraph that are matched by holes in the redex). This decomposition is then reassembled with the reactum R' replacing the matched portion of G, with select parts of d substituted into the holes of R', forming the resulting bigraph G'.*

$$G = C \circ R.d \to C \circ R'.\eta(d) = G'$$

We require further that the context C be active*, that is, that every control above holes of C are active (see CCS example below).*

We have suppressed details of the handling of names here by using the notation "$R.d$"; we have also suppressed details in the phrase "with select parts of d" and not explained the use of the map η. We refer the reader to [21] or [19] for details. The present paper can be read without understanding these details, as reaction in our examples always take the form of the following special case:

$$a = C \circ R \circ d \to C \circ R' \circ d\ .$$

Definition 7 (Bigraphical Reactive System)**.** *We use the notation $BG(\Sigma,\mathscr{R})$ to denote a bigraphical reactive system with a* signature Σ *(the set of constituent controls), and a set of reaction rules $\mathscr{R}$. More formally, $BG(\Sigma,\mathscr{R})$ is an* spm category *[21] in which the objects are interfaces and the arrows are bigraphs (which we refer to as* agents *of $BG(\Sigma,\mathscr{R})$), equipped with a set of reaction rules $\mathscr{R}$.*

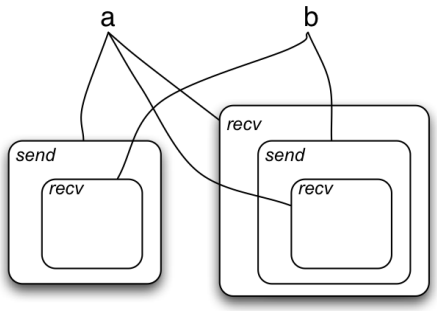


Figure 5: The process $\mathsf{send}(a).\mathsf{recv}(b).\mathsf{nil} \mid \mathsf{recv}(a).\mathsf{send}(b).\mathsf{recv}(a).\mathsf{nil}$

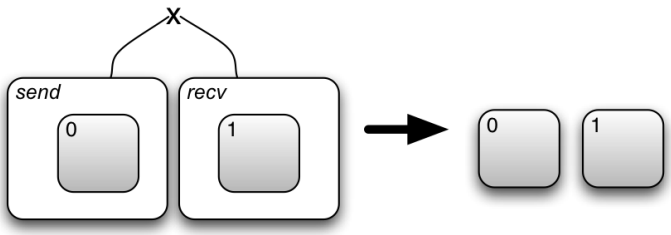


Figure 6: The R_{CCS} reaction rule

As an example, we introduce a very simple calculus in the style of the Calculus of Communicating Systems (CCS) [17], where we first give an encoding of the terms as bigraphs, and then define a reaction rule that imbues these terms with dynamic behaviour. Interested readers are referred to [21] for a real encoding of CCS.

Our calculus defines sequencing ($t.P$), parallel composition ($t \mid t$), and sending and receiving on a named channel ("x!" and "y?", respectively, where x and y are channel names). The encoding of these constructs into the bigraphical term language in Definition 5 is straightforward — these primitives are already defined in terms of the bigraphical term language, except for "send" and "receive" which we straightforwardly encode as nodes with controls send and recv, each with arity 1. Fig. 5 gives a graphical representation of the process $\mathsf{send}(a).\mathsf{recv}(b).\mathsf{nil} \mid \mathsf{recv}(a).\mathsf{send}(b).\mathsf{recv}(a).\mathsf{nil}$. According to our encoding, sequencing is represented by prefixing, parallel composition by juxtaposition, actions (such as send and recv) by *passive* controls, and channels by outer names. This is by no means the only encoding possible, but this technique is one of the most straightforward.

Having developed the encoding of our calculus within bigraphs, we can give a reaction rule R_{CCS} that will (through repeated rewriting) reduce the term as far as possible based upon agreement between parallel processes as to which action should be taken next:

$$R_{CCS} \stackrel{\text{def}}{=} \mathsf{recv}(x).-_0 \mid \mathsf{send}(x).-_1 \to -_0 \mid -_1$$

This rule is presented graphically in Fig. 6. It essentially "peels off" the outer layers of the terms where a send and a recv action are linked to the same channel name, rewriting the entire bigraph to the juxtaposition of whatever was nested inside those send and recv controls (i.e. the parts of the bigraph matched by the holes in the redex). As an example, the CCS reaction $a!.b? \mid a?.c! \to b? \mid c!$ becomes the

bigraphical reaction

$$\mathsf{send}(a).\mathsf{recv}(b).\mathsf{nil} \mid \mathsf{recv}(a).\mathsf{send}(c).\mathsf{nil} \to \mathsf{recv}(b).\mathsf{nil} \mid \mathsf{send}(c).\mathsf{nil}$$

3 Example

Aside from their role as a meta-calculus for the study of process modelling formalisms, bigraphical reactive systems are intended to provide a basis upon which to construct models of the kinds of context-aware and ubiquitous systems that are becoming increasingly popular. Consequently, we introduce an example based on modelling a context-aware social network notification system, such that a user is notified whenever a friend is in the same physical location.

We will give this example without using the link-graph part of bigraphs to keep it simple. We emphasise that the example generalises to a more interesting one in which connectivity counts — where notification is dependent not only on physical co-location but also on whether or not users and friends are virtually connected through their laptops and phones.

We will subsequently extend this to a system in which not all friends, but rather only particular designated "special friends", trigger notifications, and show that (and in what sense) the latter system is a refinement of the former.

The example system captures the dynamics of some physical environment (consisting of discrete zones within which we can detect the presence of a user by some mechanism that is outside the scope of this model) in which a user's friends move from zone to zone. When one of the user's friends is present in the same zone as the user, a notification is given, modelled by adding a "notification" node to the zone.

3.1 The abstract system: BRS_{notify}

We first define controls Z (Zone), U (User), F (Friend), N (Notification) and S (Special friend marker). Every control has arity 0 and every control is active; altogether we have a signature

$$\Sigma_N = \mathsf{Z}, \mathsf{U}, \mathsf{F}, \mathsf{N}$$

The bigraphs of our systems are thus arbitrary trees over these controls. We shall of course be interested only in those where Z are inner nodes and the remaining controls are leaves.

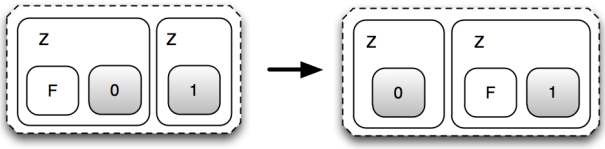
With these particular bigraphs in mind, we give reaction rules reconfiguring a bigraph by allowing nodes with control F — friends — to move between nested zones as follows. These rules are illustrated graphically in Fig. 7.

$$\begin{aligned}
M_1 &= \mathsf{Z}.(\mathsf{F} \mid -_0) \mid \mathsf{Z}.-_1 && \to && \mathsf{Z}.-_0 \mid \mathsf{Z}.(\mathsf{F} \mid -_1) \\
M_2 &= \mathsf{Z}.(\mathsf{Z}.(\mathsf{F} \mid -_0) \mid -_1) && \to && \mathsf{Z}.(\mathsf{Z}.-_0 \mid \mathsf{F} \mid -_1) \\
M_3 &= \mathsf{Z}.(\mathsf{Z}.-_0 \mid \mathsf{F} \mid -_1) && \to && \mathsf{Z}.(\mathsf{Z}.(\mathsf{F} \mid -_0) \mid -_1)
\end{aligned}$$

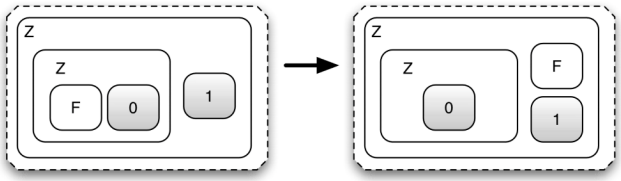
Reaction rules are here given on the form "$R \to R'$" rather than the more precise (R, R', η); recall from the above introduction to bigraphs that we use the former form whenever η is inconsequential (in this case, it is the identity map).

We extend the movement rules M with an additional rule R_1 for notifications to be issued when a U (user) and F (friend) node exist within the same zone. This reaction rule is illustrated in Fig. 8.

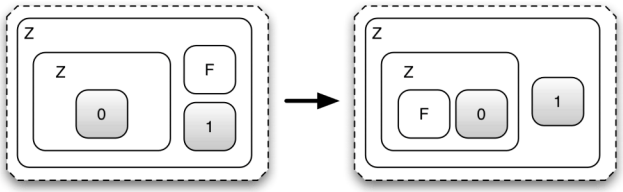
$$\begin{aligned}
&\Sigma_N = \Sigma_M \cup \{\mathsf{U}, \mathsf{N}\} \\
&R_1 = \mathsf{Z}.(\mathsf{U} \mid \mathsf{F} \mid -_0) \qquad \to \qquad \mathsf{Z}.(\mathsf{U} \mid \mathsf{F} \mid \mathsf{N} \mid -_0)
\end{aligned}$$



(a) M_1



(b) M_2



(c) M_3

Figure 7: Reaction rules M_1, M_2 and M_3 that allow $friend$ nodes to move between *zones*.

Let BRS_{notify} be the bigraphical reactive system formed by the addition of the reaction rule R_1 to the set of movement rules M:

$$BRS_{notify} = BG(\Sigma_N, M \cup \{R_1\})$$

3.2 The concrete system: $BRS_{selective}$

We now create a second bigraphical reactive system, this one refining (both intuitively and in a sense to be made precise) the system BRS_{notify} just introduced. In this new system, instead of simply notifying whenever *any* friend is present in the same zone as the user, we wish only to issue a notification in the

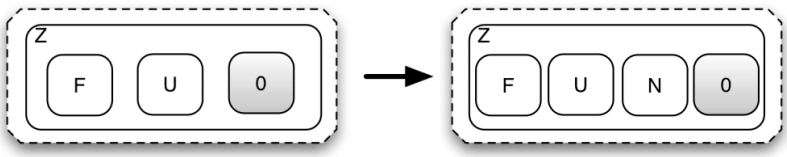


Figure 8: Reaction rule R_1

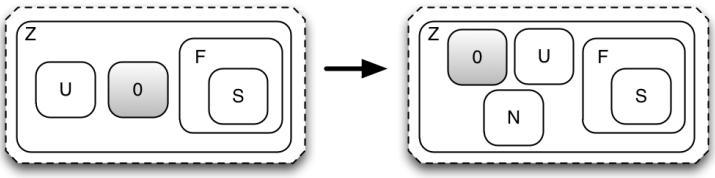


Figure 9: Reaction rule R_2

presence of a particular designated friend, distinguished by the presence of an S (special friend marker) inside the friend node in question. Consequently, we define the set of controls Σ_S for $BRS_{selective}$ to include (in addition to the controls of Σ_N) the S control. The modified reaction rule R_2 is presented graphically in Fig. 9.

$$\begin{aligned} \Sigma_S &= \Sigma_N \cup \{\mathsf{S}\} \\ R_2 &= \mathsf{Z}.(\mathsf{U} \mid \mathsf{F}.\mathsf{S} \mid -_0) \to \mathsf{Z}.(\mathsf{U} \mid \mathsf{F}.\mathsf{S} \mid \mathsf{N} \mid -_0) \\ BRS_{selective} &= BG(\Sigma_S, M \cup \{R_2\}) \end{aligned}$$

At an intuitive level, this BRS refines the one of the previous sub-section. In the following section, we shall define exactly in what sense this is the case.

4 Vertical BRS Refinement

We recall the distinction here between *horizontal* and *vertical* refinement. Vertical refinement is concerned with moving between differing levels of abstraction, or indeed completely independent modelling languages, whereas horizontal refinement instead aims to relate models specified at the same fundamental level of abstraction, and within the same modelling setting. When we refer to the refinement of a BRS, we mean a vertical refinement, indeed, this is the only meaningful interpretation, as a BRS is the category consisting of (infinitely) many actual agents of the same general shape. We will later return (briefly) to what it would mean for an *agent* to be refined, that is, to a horizontal refinement between two agents of the same BRS (each of which would be bigraphs, representing — for example — two CCS processes).

To summarise the distinction between horizontal and vertical refinement in the setting of BRSs: In the former case, we are talking about what we can observe of all such agents, whereas in the latter we are referring to what we can observe of the behaviour of a single agent. In the present section, we consider vertical refinement; we comment on horizontal refinement in the subsequent section.

4.1 Safe refinements

First, what observations can you make of bigraphical agents? While the notion of a trace is familiar within refinement literature, within bigraphical reactive systems it is unclear exactly what might correspond to an *action* within the usual definition of a trace. Consequently, we formulate a trace of a BRS such that each element of the trace is a bigraphical agent (i.e., a bigraph of that BRS). Therefore the notion of trace is not one of a system exhibiting behaviour in the form of some observable actions, rather, it is the entire state of the model as it changes over time such that every element of the trace is a bigraph, related to the next element of the trace by the application of some reaction rule. While this may seem

very crude at first glance, it is important to remember that the dynamic behaviour of a bigraph is derived from reaction rules and the structure in a perhaps more direct manner than in many other calculi. As such, it makes sense to consider the abstract specification to comprise, by itself, an entire observation — cf. the structure of agents of BRS_{notify} above.

If an observation is a complete agent of the abstract specification, what then is an observation of an agent of the concrete system? We leave that to the system constructor, merely insisting that the observations one makes of concrete implementation agents must somehow be a function of their structure. Thus, observations of concrete agents are given by a structure-preserving map from concrete agents to abstract ones. In the parlance of category theory, this is called a "functor", a functor that we shall in this instance call an *abstraction functor*.

Definition 8 (Trace, observation)**.** *For a given BRS A, a* trace *is a (possibly infinite) sequence of bigraphs (agents) $\langle a_1, a_2, \ldots \rangle$, such that for each a_i and a_{i+1} in the sequence there is a reaction $a_i \to a_{i+1}$. If $s = \langle s_1, \ldots, s_n \rangle$ and $t = \langle t_1, \ldots \rangle$ are traces and $s_n \to t_1$, we may form the* composite trace *$s;t = \langle s_1, \ldots, s_n, t_1, \ldots \rangle$. In this case we say that t is an* extension *of s. We write $Tr(A)$ for the set of all traces of a given BRS A. If $F : A \to A'$ is a functor and $\langle a_1, a_2, \ldots \rangle \in Tr(A)$ is a trace of A, we apply F pointwise to obtain a trace $F(t) = \langle F(a_1), F(a_2), \ldots \rangle$.*

Note that $Tr(x)$ is by definition prefix-closed; that is, for any trace $t \in Tr(x)$, every prefix t' of t is also in $Tr(x)$.

Of course, not just any functor will do: to have a refinement, the dynamic behaviour of the concrete implementation must be allowed by the dynamic behaviour the abstract specification allows on its agents, the observations. Altogether, our notion of refinement follows from the usual trace equality, however, because a BRS tends to permit too much observation, our bigraphical notion of refinement requires as a side condition that there exist an abstraction functor $F : C \to A$ such that for any trace $\langle c_0, c_1, \ldots \rangle$, F gives rise to a trace $\langle F(c_0), F(c_1), \ldots \rangle$. We present vertical refinement as the conjunction of two constituent definitions, separating the preservation of orthogonal safety and liveness properties through refinement.

Definition 9 (Safe Vertical Refinement)**.**

$$A \overset{\text{safe}}{\sqsubseteq}_F C \quad \overset{\text{def}}{=} \quad F(Tr(C)) \subseteq Tr(A)$$

This definition satisfies the "reduction of nondeterminism" role of refinement, in that it is always valid to simply pick one alternative and implement it in C when presented with nondeterministic choice in A.

Lemma 1. *Safe Vertical Refinement is transitive and reflexive for the identity functor.*

Proof. Reflexivity is trivial. Suppose $A \overset{\text{safe}}{\sqsubseteq}_F C$ and $C \overset{\text{safe}}{\sqsubseteq}_G D$. Then $FG(Tr(D)) \subseteq F(Tr(C)) \subseteq Tr(A)$. □

We proceed to illustrate safe refinement using the two BRSs above, then give a sufficient condition for an abstraction functor to yield a safe refinement.

Recall our claim that $BRS_{selective}$, which issues notifications upon co-location with "special friends" is a refinement of BRS_{notify}, which does so upon co-location with any friend. The latter employs an additional control S. This indicates that our abstraction functor must (at the very least) ensure that all nodes of control S must be hidden, renamed or removed so as to ensure that the codomain of F is BRS_{notify} (i.e. that F can transform any agent of $BRS_{selective}$ into a valid agent of BRS_{notify}).

By this reasoning, we arrive at an abstraction functor "pattern" that is likely applicable to many other BRSs. We call this the *hiding functor*. Its essential function is to simply hide, for a given signature Σ,

all nodes that have been assigned controls from some particular set of controls H. This includes joining any children of nodes that will be hidden to parents that will remain visible after the application of the hiding functor. For our example, the hiding set $H = \{\mathsf{S}\}$ (i.e. the designated "special" friend control).

Definition 10 (Hiding Functor)**.** *We define an abstraction functor $F_{\Sigma,H} : BG(\Sigma) \to BG(\Sigma \setminus H)$ for hiding, parametrised by Σ, the signature of the "implementation" BRS, and H, a set of controls to be hidden. On* objects*, this functor is the identity. On* arrows*, its action is $F_{\Sigma,H}((V,E,prnt,ctrl,link)) \stackrel{\text{def}}{=} (V',E,ctrl',prnt',link)$, where*

- $V' = \{v \in V | ctrl(v) \notin H\}$
- $ctrl' = ctrl \restriction V'$, *and*
- $prnt'(l) \stackrel{\text{def}}{=} \begin{cases} prnt(l) & \text{where } ctrl(prnt(l)) \notin H \\ prnt'(prnt(l)) & \text{otherwise} \end{cases}$

This "hiding functor" is an abstraction functor for our example system. Recalling the definition of a bigraphical agent (and therefore of an arrow in the category BRS_{notify} or $BRS_{selective}$) given in Definition 2, the purpose of this hiding functor is to exclude any nodes that have a control that is in the set of hidden controls H, exclude these controls from the control map $ctrl$, and recursively recreate the parent map $prnt$ such that any children of a node with a control in H is attached to its most immediate place-graph ancestor that is not marked with a control in H. We call the abstraction functor for our example notification system A_{friend}, which is defined as the hiding functor above, instantiated with $H = \{\mathsf{S}\}$.

While the hiding functor has the flavour of a forgetful functor — it dispenses with structure — it cannot reasonably be called so as it is not faithful. Many distinct configurations (e.g. special-friend controls) will map to the same bigraph. This is a technical distinction only; we use "hiding" in no special sense, except as a name for abstraction functors of this general shape.

It is easy to prove that with A_{friend} as abstraction functor, $BRS_{selective}$ is indeed a safe refinement of BRS_{notify}. However, instead of proving so directly, we shall instead provide a general theorem about abstraction functors: When they preserve reaction, and in particular, when they preserve just reaction *rules*, they give rise to safe refinement.

Theorem 1. *Let $F : C \to A$ be an abstraction functor. If F preserves reaction, that is, if $c \to c'$ implies $F(c) \to F(c')$, then $A \stackrel{\text{safe}}{\sqsubseteq}_F C$.*

Proof. Immediate from Definition 9 of safe refinement. □

From this theorem it becomes apparent that an abstraction functor may be any functor at all that obeys this property.

The terminology deceives, here: The guarantee that the concrete system has *no more* behaviour than the abstract one is in fact upheld by the abstraction functor *preserving* behaviour.

Of course, proving that a functor preserves reaction need not at all be easy. Fortunately, we can exploit the connection between static structure and dynamic behaviour of bigraphs: a functor which preserves the reaction *rules*, structurally, will also preserve (dynamic) reaction, and will thus be a safe refinement.

Theorem 2 (Safe Abstraction Functors)**.** *Let $A = BG(\Sigma,\mathscr{R})$ and $C = BG(\Sigma',\mathscr{R}')$ be BRSs. A functor $F : C \to A$ yields a safe vertical refinement $A \stackrel{\text{safe}}{\sqsubseteq}_F C$* if *it satisfies the following conditions.*

1. *It preserves and respects tensor.*
2. *It preserves active contexts.*

3. *It preserves reaction rules: For any reaction rule* $(R,R',\rho) \in \mathscr{R}'$ *(a) the* F*-image* $(F(R),F(R'),\rho)$ *is a rule in* $\mathscr{R}$*; and (b) for any parameter* d *of that rule,* $\overline{\rho}(F(d)) = F(\overline{\rho}(d))$.

Proof. Suppose $c_1,\ldots,c_n$ is a trace of C. It is sufficient to prove that for each $i < n$, there is a reaction $F(c_i) \to F(c_{i+1})$. We know that $c_i \to c_{i+1}$, so there is some reaction rule $(R,R',\rho) \in \mathscr{R}'$, context E of C, and some set of names Z s.t.

$$c_i = E \circ (R \otimes 1_Z) \circ d \qquad \to \qquad E \circ (R' \otimes 1_Z) \circ \overline{\rho}(d) = c_i'$$

Where $\overline{\rho}(d)$ is the instantiation of parameters (see [21] for details). But then, because $(F(R),F(R'),\rho)$ is a rule of $\mathscr{R}$, we compute and find $a_i = F(c_i) = F(E \circ (R \otimes 1_Z) \circ d) = F(E) \circ (F(R) \otimes 1_{F(\mathsf{Z})}) \circ F(d) \to F(E) \circ (F(R') \otimes 1_{F(\mathsf{Z})}) \circ \overline{\rho}(F(d)) = F(E) \circ (F(R') \otimes 1_{F(\mathsf{Z})}) \circ F(\overline{\rho}(d)) = F(E \circ (R' \otimes 1_Z) \circ \overline{\rho}(d)) = F(c_i') = a_i'$ □

We remark that the three conditions of this Theorem appear to be good candidates for a definition of a morphism of *parametric* reactive systems, as suggested in the forthcoming [7].

It is straightforward to verify that for our example BRSs, $BRS_{selective}$ and BRS_{notify}, the hiding functor does in fact satisfy the three conditions of this Theorem. Thus we have the following corollary:

Corollary 1. $BRS_{selective}$ *is a sound refinement of* BRS_{notify} *with respect to the abstraction functor* A_{friend}*, that is,*

$$BRS_{notify} \overset{\text{safe}}{\sqsubseteq}_{A_{friend}} BRS_{selective}$$

The $\overset{\text{safe}}{\sqsubseteq}$ relation captures safety properties of the system being refined (i.e. it does not permit a refined model any undesirable extra behaviour, provided that the abstraction functor does not hide any "undesirable" behaviour). However, it does not guarantee that the system does anything at all (i.e. an empty trace is a safe refinement of any system). To guarantee that some additional liveness properties are preserved by refinement, it is necessary to extend our definition.

4.2 Live refinements

In order to guarantee that a given concrete system actually exhibits any of the desirable behaviour of the abstract system that it refines, we must define a notion of liveness. Whereas in a process algebraic setting it might be possible to rely on the presence of a particular output (or all possible outputs) to define "desired" observable behaviour, within a bigraphical setting the lack of any primitive notions of "input" or "output" (it is up to the system designer to define what these concepts mean with respect to a particular model) means that it is necessary to explicitly choose such "desirable" behaviours.

In the absence of an intrinsic notion of desirable behaviour, we further parametrise our notion of liveness, already parametric in terms of the abstraction functor F, on the admissible traces. This parametrisation on the notion of admissibility is akin to those used in [13, 11].

Definition 11 (Live Vertical Refinement)**.** *Let* $F : C \to A$ *be an abstraction functor, let* $\mathbf{C} \subseteq Tr(C)$ *be the* admissible traces *for* C*, and let similarly* $\mathbf{A} \subseteq Tr(A)$*, the admissible traces of* A*. We then say that* $(C,\mathbf{C})$ *is a* live *refinement of* $(A,\mathbf{A})$ *iff for every trace* s *of* $Tr(C)$*, whenever* $F(s)$ *has an extension* t' *to an admissible trace* $F(s);t' \in \mathbf{A}$*, then there exists an extension* s' *of* s *to an admissible trace* $s;s' \in \mathbf{A}$ *with* $F(s') = F(t')$*. In this case we write:*

$$(A,\mathbf{A}) \overset{\text{live}}{\sqsubseteq}_F (C,\mathbf{C})\ .$$

If we wish to take the admissible traces $\mathbf{A}$ of the abstract system A as canonical, we can define $\mathbf{C}$ as those traces whose F-images are admissible.

Lemma 2. *Live Vertical Refinement is transitive.*

Proof. Suppose $(A,\mathbf{A}) \overset{\text{live}}{\sqsubseteq}_F (C,\mathbf{C})$ and $(C,\mathbf{C}) \overset{\text{live}}{\sqsubseteq}_G (D,\mathbf{D})$, and suppose $FG(t);u' \in \mathbf{A}$. Then u' has a pre-image s' with $G(t);s' \in \mathbf{C}$; but then s' has a pre-image t' with $t;t' \in \mathbf{D}$. □

Let us provide a suitable set of admissible traces for our running example, BRS_{notify}. For this BRS, the obvious notion of admissibility (think "successful") is when notification has occurred. So we define the set of admissible traces as simply those finite traces in which the user has been notified, that is, in which the final agent contains the notification control next to the user and his friend:

$$\mathbf{S}_{notified} \overset{\text{def}}{=} \{\langle a_1,\ldots,a_n\rangle \in Tr(BRS_{notify}) \mid \exists C.\ a_n = C \circ (\mathsf{U} \mid \mathsf{F} \mid \mathsf{N})\}$$

For $BRS_{selective}$, we transfer the notion of admissiblity:

$$\mathbf{S}_{selective} \overset{\text{def}}{=} \{t \in Tr(BRS_{notify}) \mid F(t) \in \mathbf{S}_{notified}\}$$

The selective system $BRS_{selective}$ under these notions of admissibility is in fact *not* a live refinement of the original one BRS_{notify}. One might think so: After all, one can extend a trace to admissibility simply by moving the special friend next to the user. Unfortunately, there need not be a special friend, and even if there were, the abstract system might extend to admissibility by moving a (non-special) friend next to the user. We will now show this in detail, thus proving of the following proposition:

Proposition 1. $(BRS_{notify},\mathbf{S}_{notify}) \overset{\text{live}}{\not\sqsubseteq}_{A_{friend}} (BRS_{selective},\mathbf{S}_{selective})$.

Proof. Consider an agent $\mathsf{Z}.(\mathsf{U} \mid \mathsf{F})$ of $BRS_{selective}$. Applying A_{friend} we find simply $A_{friend}(\mathsf{Z}.(\mathsf{U} \mid \mathsf{F})) = \mathsf{Z}.(\mathsf{U} \mid \mathsf{F})$, which succeeds after just one reaction

$$\mathsf{Z}.(\mathsf{U} \mid \mathsf{F}) \to \mathsf{Z}.(\mathsf{U} \mid \mathsf{F} \mid \mathsf{N})$$

by reaction rule R_1. Now, if we actually had a live refinement, we should be able to match this reaction in $BRS_{selective}$. A simple inspection of the rules however prove that this is not possible. □

This is, however, not a show-stopper, rather it is a welcome demonstration of the utility of such a vertical refinement mechanism. We could remedy this situation by introducing into $BRS_{selective}$ an additional reaction rule that spontaneously adds the designated friend marker S to any friend F. However, this seems to contradict the intuition of the model, so in this instance it is perhaps better to leave $BRS_{selective}$ unmodified and accept that there are (known) conditions under which this BRS cannot progress to a successful state.

Having defined our two separate (live and safe) refinement relations, we can complete the definition of safe and live vertical refinement:

Definition 12 (Safe and Live Vertical Refinement)**.**

$$(A,\mathbf{A}) \sqsubseteq_F (C,\mathbf{C}) \overset{\text{def}}{=} A \overset{\text{safe}}{\sqsubseteq}_F C \wedge (A,\mathbf{A}) \overset{\text{live}}{\sqsubseteq}_F (C,\mathbf{C})$$

5 Discussion & related work

Having introduced our notion of vertical BRS refinement and shown the conditions under which it is safe and live with respect to the chosen abstraction functor, we now discuss potential approaches to horizontal refinement and related work. As it happens, both topics take us to the general refinement of Reeves and Streader [23, 24].

General horizontal refinement recognises three components to refinement: *entities* E, i.e., the specifications and implementations being refined; *contexts* Ξ, which are the environment within which the entities interact; and a *user*, which defines the possible observations $O(-)$ that can be made of an entity within a particular context. Refinement is then the relation

$$A \sqsubseteq_{\Xi,O} C \stackrel{\text{def}}{=} \forall x \in \Xi . O([C]_x) \subseteq O([A]_x) \ ,$$

where Ξ is the set of contexts, O is a map assigning observations to entities in contexts, and $[-]_x$ inserts an entity into context x.

Interestingly, our proposed notion of bigraphical *vertical* refinement falls under the umbrella of general *horizontal* refinement. Entities would be BRSs (like BRS_{notify} and $BRS_{selective}$); contexts Ξ would be just the trivial context, which leaves the entity unchanged. Finally, the observation map O is in our case simply $Tr(-)$, the map that takes a BRS to the traces observable of it. We do not think this is a coincidence. It seems intuitive that horizontal refinement of an entire class of agents would correspond to vertical refinement.

What about general *vertical* refinement, then? The definition of vertical refinement within the general refinement framework [24] relies upon a notion of *layers*, representing a level of abstraction in terms of (E_L, Ξ_L, O_L), where E_L is a set of entities, Ξ_L is a set of contexts and O_L is an observation function. Vertical refinement is then defined in terms of a Galois-connection that interprets high-level entities as low-level ones and vice versa.

The analogy of this notion with our use of an abstraction functor $F : C \to A$ should be apparent: If we could find that functor F to be one of an adjoint pair, we would be in an analogous situation. Unfortunately, it remains unclear if such an adjunction would retain the intuition behind the Galois-connection of general vertical refinement: morphisms (i.e., bigraphs) do not measure approximation; they represent the agents under investigation. In particular, the hiding functors used for the example in the present paper do not appear to be part of adjoint pairs.

Leaving vertical refinement behind, what is then a good notion of horizontal refinement for bigraphs? Returning to general horizontal refinement, bigraphs actually do come with a notion of entity, context, and observation, namely agents (roughly, bigraphs with no holes/inner names), bigraph contexts (bigraphs with holes/inner names), and an LTS (given a BRS). We have in the present paper by-passed the LTS as the notion of observation, following the bigraphical connection by structure and dynamics to its extreme conclusion, using the structure of the abstract specification as the observations.

For horizontal refinement, this approach appears not sensible: We would after all be relating agents of the same BRS. Important examples (like CCS-process refinement) cannot be expressed within this particular approach, which should guide the development of other horizontal refinement strategies for bigraphical agents. One obvious choice seems now to be the LTS intrinsic to BRSs. We have yet to pursue this option; we caution that while BRS LTSs have been successful in recovering semantics of various process algebras and other models of concurrency, it has been less successful in providing useful semantics for pervasive systems, one of our key interests.

However, even leaving the question of suitable observations open, we would likely find a notion

inside general horizontal refinement by taking

$$a \sqsubseteq_O c \stackrel{\text{def}}{=} \forall x \in \Xi . O(x \circ c) \subseteq O(x \circ a) \ ,$$

where a and c are agents of some BRS B; Ξ is the set of contexts of that BRS, and O is some notion of the semantics of agents of B, perhaps traces of the LTS of B, or perhaps some other notion. Indeed, early indications are that this approach would be promising in recovering (for example) CCS process refinement, contingent upon an appropriate notion of observation.

5.1 Related Work

Restricting the set of controls admissible under a certain control, or requiring a control to be present is well-studied in bigraphs (e.g., [2, 21, 19, 22]). However, that study has invariably focused on ensuring that the bigraphical LTS theory is retained under such additional constraints, and are thus only superficially related to the present paper.

Goldsmith & Creese [9] explore an approach to refinement within bigraphs (and particularly within Spygraphs, a specialisation of bigraphs). They observe the ease with which one may derive an LTS for a BRS that is labeled exclusively by the trivial context *id* (equivalent to a τ action in a process algebraic setting). These kinds of contextual labels are not helpful for analysis, as they capture no behaviour. Similarly, the LTS semantics of bigraphs share the same intentionality inherent in the graphical presentation. While Goldsmith & Creese suggest (to good effect in a CSP setting) that it may be appropriate to perform hiding at a process-level before considering a transition into bigraphs, this would seem inappropriate for many modelling situations (e.g., those which have no convenient term or process representation). While the transformation on bigraphical reactive systems proposed by that work may give rise to a refinement that is appropriate for some situations, we aim instead in this present work to work directly within the structure of bigraphs so as to ensure generality. As bigraphs attempt to be both a modelling formalism and a general meta-calculus for existing process calculi, it seems appropropriate that the notion of refinement we introduce should be similarly general, in the hope that we may recover calculus-specific notions of refinement within this general setting.

6 Conclusion

We have presented a vertical refinement mechanism for bigraphical reactive systems that adds refinement to the toolbox of model builders working within a bigraphical setting. The addition of a sufficient condition for safe abstraction functors, and the accompanying observation that it is the *preservation* of behaviour with respect to reaction that guarantees that a refinement exhibits no undesirable behaviour, provides a firm foundation from which to explore the limits and utility of this kind of vertical refinement.

We have pointed out a clear connection to the existing work on generalising refinement across many modelling formalisms, and therefore it seems appropriate (given the application of BRSs as a *meta-calculus*) that our notion of vertical refinement is also in some sense general. We leave for future work the exploration of further mechanisms for horizontal refinement within a bigraphical setting, noting that such a notion would very likely fall within the model of general refinement, and thus likely generalise well to other modelling formalisms encoded within bigraphical reactive systems.

References

[1] L. Birkedal, S. Debois, E. Elsborg, T. Hildebrandt & H. Niss (2006): *Bigraphical models of context-aware systems*. In: *Foundations of Software Science and Computation Structures*, Springer, pp. 187–201, doi:10.1007/11690634_13.

[2] L. Birkedal, S. Debois & T. Hildebrandt (2008): *On the construction of sorted reactive systems*. In: *CONCUR 2008*, Springer, pp. 218–232, doi:10.1007/978-3-540-85361-9_20.

[3] T. Bolusset & F. Oquendo (2002): *Formal refinement of software architectures based on rewriting logic*. In: *International Workshop on Refinement of Critical Systems: Methods, Tools and Experience, Grenoble*.

[4] M. Bundgaard & V. Sassone (2006): *Typed polyadic pi-calculus in bigraphs*. In: *Proceedings of the 8th ACM SIGPLAN international conference on Principles and practice of declarative programming*, ACM, pp. 1–12, doi:10.1145/1140335.1140336.

[5] T.C. Damgaard, V. Danos & J. Krivine (2008): *A Language for the Cell*. Technical Report TR-2008-116, IT University of Copenhagen.

[6] T.C. Damgaard & J. Krivine (2008): *A Generic Language for Biological Systems based on Bigraphs*. Technical Report TR-2008-115, IT University of Copenhagen.

[7] S. Debois: *Computation in the Informatic Jungle*. To appear. Draft available at `http://www.itu.dk/people/debois/pubs/computation.pdf`.

[8] E. Elsborg, T. Hildebrandt & D. Sangiorgi (2009): *Type Systems for Bigraphs*. In Christos Kaklamanis & Flemming Nielson, editors: *Proceedings of the 4th International Symposium on Trustworthy Global Computing (TGC 2008)*, *Lecture Notes in Computer Science* 5474, Springer-Verlag, pp. 126–140, doi:10.1007/978-3-642-00945-7_8.

[9] M. Goldsmith & S. Creese (2010): *Refinement-Friendly Bigraphs and Spygraphs*. In: *2010 8th IEEE International Conference on Software Engineering and Formal Methods*, IEEE, pp. 203–207, doi:10.1109/SEFM.2010.25. Available at `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5637430`.

[10] D. Grohmann & M. Miculan (2007): *Reactive Systems over Directed Bigraphs*. In Luís Caires & Vasco Thudichum Vasconcelos, editors: *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07)*, *Lecture Notes in Computer Science* 4703, Springer-Verlag, pp. 380–394, doi:10.1007/978-3-540-74407-8_26.

[11] M. Hennessy & C. Stirling (1985): *The power of the future perfect in program logics*. *Information and Control* , pp. 23–52.

[12] T. Hildebrandt, H. Niss & M. Olsen (2006): *Formalising Business Process Execution with Bigraphs and Reactive XML*. In Paolo Ciancarini & Herbert Wiklicky, editors: *Proceedings of the 8th International Conference on Coordination Models and Languages (COORDINATION'06)*, *Lecture Notes in Computer Science* 4038, Springer-Verlag, pp. 113–129, doi:10.1007/11767954_8.

[13] T.T. Hildebrandt (1999): *Categorical Models for Fairness and a Fully Abstract Presheaf Semantics of SCCS with Finite Delay*. In: *CTCS'99*, LNCS, doi:10.1016/S1571-0661(05)80311-1.

[14] O.H. Jensen (2006): *Mobile Processes in Bigraphs*. Available at `http://www.cl.cam.ac.uk/~rm135/Jensen-monograph.pdf`.

[15] J. Krivine, R. Milner & A. Troina (2008): *Stochastic bigraphs*. *Electronic Notes in Theoretical Computer Science* 218, pp. 73–96, doi:10.1016/j.entcs.2008.10.006.

[16] J. Leifer & R. Milner (2006): *Transition systems, link graphs and Petri nets*. *Journal of Mathematical Structures in Computer Science* 16(6), pp. 989–1047, doi:10.1017/S0960129506005664.

[17] R. Milner (1980): *A calculus of communicating systems*. Springer-Verlag.

[18] R. Milner (2005): *Axioms for Bigraphical Structure*. *Journal of Mathematical Structures in Computer Science* 15(6), pp. 1005–1032, doi:10.1017/S0960129505004809.

[19] R. Milner (2006): *Pure Bigraphs: Structure and Dynamics*. Information and Computation 204(1), pp. 60–122, doi:10.1016/j.ic.2005.07.003.

[20] R. Milner (2007): *Local Bigraphs and Confluence: Two Conjectures: (Extended Abstract)*. In Roberto Amadio & Iain Phillips, editors: *Proceedings of the 13th International Workshop on Expressiveness in Concurrency (EXPRESS 2006)*, *Electronic Notes in Theoretical Computer Science* 175, Elsevier, doi:10.1016/j.entcs.2006.07.035.

[21] R. Milner (2009): *The space and motion of communicating agents*. Cambridge University Press.

[22] S. Ó Conchúir (2009): *Kind Bigraphs*. In Anthony Seda, Menouer Boubekeur, Ted Hurley, Micheal Mac an Airchinnigh, Michel Schellekens & Glenn Strong, editors: *Proceedings of the Irish Conference on the Mathematical Foundations of Computer Science and Information Technology (MFCSIT 2006)*, *Electronic Notes in Theoretical Computer Science* 225, Elsevier, pp. 361–377, doi:10.1016/j.entcs.2008.12.086.

[23] S. Reeves & D. Streader (2008): *General refinement, part one: interfaces, determinism and special refinement*. *Electronic Notes in Theoretical Computer Science* 214, pp. 277–307, doi:10.1016/j.entcs.2008.06.013.

[24] S. Reeves & D. Streader (2008): *General refinement, part two: flexible refinement*. *Electronic Notes in Theoretical Computer Science* 214, pp. 309–329, doi:10.1016/j.entcs.2008.06.014.

[25] M. Zhang, L. Shi, L. Zhu, Y. Wang, L. Feng & G. Pu (2008): *A Bigraphical Model of WSBPEL*. In: *Second Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering (TASE'08)*, IEEE Computer Society, pp. 117–120.